

## Problema A - Bands

Codul de [aici](#)<sup>1</sup> ar trebui să genereze (atunci când de la tastatură se citește caracterul „l”) o listă de trupe într-o ordine pseudo-aleatoare.

Cu toate acestea, rularea programului pe un sistem Linux cu sistem de protecție a memoriei generează o eroare la execuție.

Faceți modificările necesare, astfel încât programul să funcționeze corect.

În sensul rezolvării problemei nu se permite:

- modificarea comportamentului așteptat al programului;
- schimbarea seed-ului oferit funcției srand();
- restricționarea comportamentului, astfel încât anumite trupe să nu poată fi afișate.

### Precizări:

**Problema se va compila folosind compilatorul gcc.**

Rezolvarea trebuie să conțină un număr minim de modificări.

Eliminarea unui număr de trupe din lista nu este considerată o soluție validă.

### Examples:

Input	Output
l q	Here lies the biggest rock band database Press L to list random bands from the databse Press Q to quit Phoenix Deep Purple Black Sabbath Black Sabbath Deep Purple Phoenix Led Zeppelin Deep Purple Cargo Cargo

<sup>1</sup> <https://raw.githubusercontent.com/dorinel Filip/Acadnet-2016/master/Software-Interoperability/Initial-Code/A.c>

## Problema B - Shoes

Un pantof este caracterizat prin tip (stâng/drept) și mărimea sa.

Pentru a putea face parte dintr-o pereche, doi pantofi trebuie să fie de tipuri diferite și să aibă aceeași mărime.

Se cere a se spune dacă un set de pantofi poate fi organizat (în totalitate) în perechi.

Este nevoie de un program care spune dacă o colecție de pantofi poate fi organizată (integral) în perechi.

Programul de [aici](#)<sup>2</sup> rezolvă în C++ această problemă, însă nu dă întotdeauna rezultate corecte.

Sarcina ta este să-l faci funcțional!

### Formatul de input este:

Pe prima linie un nr. natural N

Pe următoarele N linii, N perechi (numere întregi) de tipul tip (0/1) și marime (nr natural).

Programul trebuie să afișeze YES dacă toți pantofii au pereche și NO în caz contrar.

**Compilarea sursei se va face OBLIGATORIU folosind compilatorul g++.**

[1] <https://raw.githubusercontent.com/dorinel Filip/Acadnet-2016/master/Software-Interoperability/Initial-Code/B.cpp>

### Examples:

Input	Output
4 0 24 0 16 1 16 1 24	YES
6 0 24 0 16 1 16 1 24 0 10 0 10	NO

<sup>2</sup> <https://raw.githubusercontent.com/dorinel Filip/Acadnet-2016/master/Software-Interoperability/Initial-Code/B.cpp>

6	NO
0 24	
0 16	
1 16	
1 24	
0 10	
1 11	

## Problema C - Maze

În fișierul maze.cpp din [arhivă](#)<sup>3</sup> este implementat un joc text-based de tipul Hunter - Prey.

Scopul Hunter-ului este de a prinde Prey, care trebuie să evite să fie prinsă. Atât H cât și P se pot mișca o singură pătrățică la un moment de timp.

H este controlat de user (voi) de la tastatură prin tastele W, A, S, D.

Scopul task-ului este de a scrie un mini AI pentru Prey, astfel încât acesta să evite Hunter-ul când acesta este la o distanță  $\leq 5$ , în orice situație, fără să se bloca în margine/culțuri.

### Detalii implementare:

- tot codul trebuie scris în funcția "movePrey"
- se **permite** crearea/folosirea de funcții ajutoare
- nu aveți voie să modificați restul codului / restricționați funcționalitatea
- don't reinvent the wheel, make use of what you have

### Compilarea sursei:

Compilarea sursei se poate face folosind comanda **make** în directorul în care aveți sursa voastră, alături de toate celelalte fișiere din arhivă.

Înainte de a putea recompila sursa va trebui să rulați comanda make clean.

### Detalii de trimitere:

- Rezolvarea problemei se va începe plecând de la sursa de [aici](#)[1].
- Pe platformă se va trimite doar fișierul maze.cpp.
- Ca și compilator selectați opțiunea Make.

**Notă:** Pentru **dezarhivarea** arhivei puteți folosi comanda **tar xvf arhiva.tar**

---

<sup>3</sup> <https://raw.githubusercontent.com/dorinel Filip/Acadnet-2016/master/Software-Interoperability/Initial-Code/C.tar>